React Bootstrap. Modals. Alerts.

Bootstrap и React можно легко интегрировать, чтобы быстро создавать красивые и отзывчивые пользовательские интерфейсы. В этом уроке, мы рассмотрим использование самых интересных компонентов из библиотеки Bootstrap.

В этом уроке, мы рассмотрим следующие темы:

1) Установка библиотеки React Bootstrap.
2) Общие примеры использования базовых компонентов.
3) Комплексное приложение с компонентами Bootstrap 5.
4) React Bootstrap: Modals - полное взаимодействие.
5) Приложение – «Список покупок».

Установка библиотеки React Bootstrap

Официальный сайт библиотеки: https://react-bootstrap.netlify.app/docs/components

Лучший способ использовать React-Bootstrap – это скачать готовый пакет, который мы можем установить с помощью **npm** (или yarn, если хотите).

Если вы планируете настраивать Sass-файлы Bootstrap или не хотите использовать CDN для таблицы стилей, может быть полезно установить и **Vanilla Bootstrap** (https://getbootstrap.com/docs/5.3/getting-started/download/#npm).

Открываем **Terminal** и выполняем команду:

npm install react-bootstrap bootstrap

Загрузив библиотеку, мы сможем импортировать по отдельности нужные компоненты, а не всю библиотеки целиком. Это помогает значительно сократить объем кода, который вы в итоге отправляете клиенту.

К примеру, импортировать компонент **Button**, можно следующим образом:

```
import Button from 'react-bootstrap/Button';
//или так
import { Button } from 'react-bootstrap';
```

Все компоненты автоматически имеют доступ к пакетам **react-bootstrap.js** и **react-bootstrap.min.js**, которые необходимы для функционирования.

Что бы для компонентов были отображены стили, необходимо подключить файл стилей **bootstrap.min.css**:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Этого будет достаточно для работы с компонентом.

Общие примеры использования базовых компонентов

В папке **src**, создадим папку **components**, а в ней файл **BaseComponents.jsx**, со следующим содержимым:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Alert from 'react-bootstrap/Alert';

function BaseComponent() {
  return (
    <>
      {[
        'primary',
        'secondary',
        'success',
        'danger',
        'warning',
        'info',
        'light',
        'dark',
      ].map((variant) => (
        <Alert key={variant} variant={variant}>
          This is a {variant} alert—check it out!
        </Alert>
      ))}
    </>
  );
}

export default BaseComponent;
```

В этом файле, мы определили компонент, который использует библиотеку **react-bootstrap** для отображения элементов интерфейса с **Bootstrap-стилями**.

import 'bootstrap/dist/css/bootstrap.min.css' - подключает стили Bootstrap в проект. Без этого компоненты, такие как Alert, не будут корректно отображаться с нужными стилями, так как CSS классы Bootstrap не будут применяться.

import Alert from 'react-bootstrap/Alert' - загружает компонент Alert из библиотеки react-bootstrap. Alert используется для создания всплывающих оповещений (алертов) с различными стилями.

Внутри компонента определен массив с названиями различных вариантов цветов для оповещений (например, primary, secondary, success и т. д.), который перебирается с помощью метода **map**.

Для каждого элемента массива создается компонент **Alert** с динамически задаваемым вариантом оформления через проп **variant**. Этот проп определяет цвет и стиль алерта (например, синий для primary, зеленый для success и т. д.).

Внутри каждого компонента **Alert** отображается текст: "This is a {variant} alert—check it out!", где {**variant**} — это название текущего варианта.

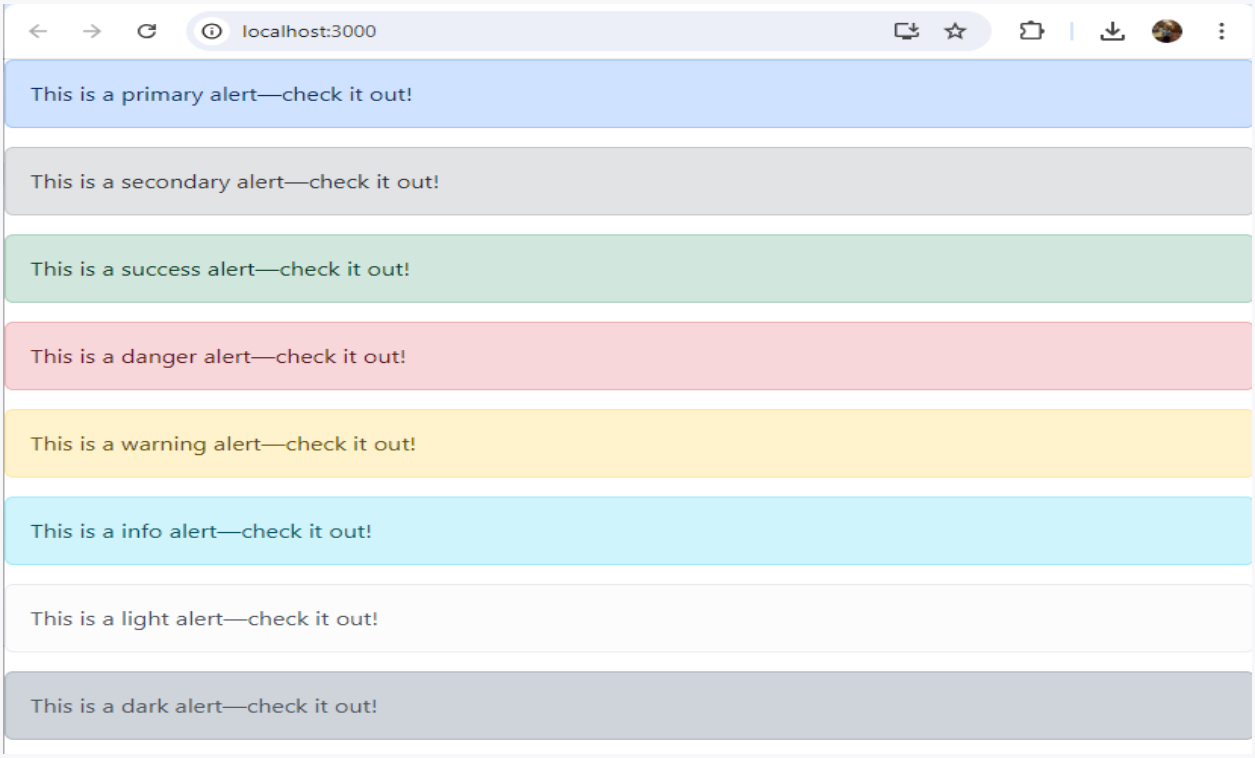
Вызовем данный компонент, в файле **App.jsx**:

```
import BaseComponent from "./components/BaseComponents";

function App() {
  return (
    <div >
      <BaseComponent />
    </div>
  );
}

export default App;
```

Запустим приложение и проверим его работу:



Давайте, в этом же компоненте определим еще пару простых компонентов bootsrap. Кнопка с бейджем. Изменим компонент **BaseComponents.jsx**:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Badge from 'react-bootstrap/Badge';
import Button from 'react-bootstrap/Button';

function BaseComponent() {
  return (
```

```
    <Button variant="primary">
      Profile <Badge bg="secondary">9</Badge>
      <span className="visually-hidden">unread messages</span>
    </Button>
  );
}

export default BaseComponent;
```

Хлебные крошки:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Breadcrumb from 'react-bootstrap/Breadcrumb';

function BaseComponent() {
  return (
    <Breadcrumb>
      <Breadcrumb.Item href="#">Home</Breadcrumb.Item>
      <Breadcrumb.Item
href="https://getbootstrap.com/docs/4.0/components/breadcrumb/">
        Library
      </Breadcrumb.Item>
      <Breadcrumb.Item active>Data</Breadcrumb.Item>
    </Breadcrumb>
  );
}

export default BaseComponent;
```

Группа кнопок:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';

function BaseComponent() {
  return (
    <ButtonGroup aria-label="Basic example">
      <Button variant="secondary">Left</Button>
      <Button variant="secondary">Middle</Button>
      <Button variant="secondary">Right</Button>
    </ButtonGroup>
  );
}

export default BaseComponent;
```

Карточка с изображением:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';

function BaseComponent() {
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Img variant="top" src="holder.js/100px180" />
      <Card.Body>
        <Card.Title>Card Title</Card.Title>
        <Card.Text>
          Some quick example text to build on the card title and make up the
          bulk of the card's content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default BaseComponent;
```

Выпадающий список:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function BaseComponent() {
  return (
    <DropdownButton id="dropdown-basic-button" title="Dropdown button">
      <Dropdown.Item href="#/action-1">Action</Dropdown.Item>
      <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
      <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
    </DropdownButton>
  );
}

export default BaseComponent;
```

Меню сайта:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Container from 'react-bootstrap/Container';
```

```
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';

function BaseComponent() {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#link">Link</Nav.Link>
            <NavDropdown title="Dropdown" id="basic-nav-dropdown">
              <NavDropdown.Item href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Item href="#action/3.3">Something</NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action/3.4">
                Separated link
              </NavDropdown.Item>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default BaseComponent;
```

Блок пагинации:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Pagination from 'react-bootstrap/Pagination';

function BaseComponent() {
  return (
    <Pagination>
      <Pagination.First />
      <Pagination.Prev />
      <Pagination.Item>{1}</Pagination.Item>
      <Pagination.Ellipsis />

      <Pagination.Item>{10}</Pagination.Item>
      <Pagination.Item>{11}</Pagination.Item>
```

```
      <Pagination.Item active>{12}</Pagination.Item>
      <Pagination.Item>{13}</Pagination.Item>
      <Pagination.Item disabled>{14}</Pagination.Item>

      <Pagination.Ellipsis />
      <Pagination.Item>{20}</Pagination.Item>
      <Pagination.Next />
      <Pagination.Last />
    </Pagination>
  );
}


export default BaseComponent;
```

Спинер анимации загрузки:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Spinner from 'react-bootstrap/Spinner';

function BasicExample() {
  return (
    <Spinner animation="border" role="status">
      <span className="visually-hidden">Loading...</span>
    </Spinner>
  );
}

export default BasicExample;
```

Множество других компонентов можно увидеть на официальном сайте:

https://react-bootstrap.netlify.app/docs/components/accordion

Давайте создадим более комплексное приложение, используя компоненты **Bootstrap 5**.

Комплексное приложение с компонентами Bootstrap 5

Создадим комплексное приложение на React, которое будет использовать компоненты из библиотеки react-bootstrap для создания адаптивного интерфейса с навигацией, карточками, таблицами и модальными окнами.

Если вы используете новый проект, не забудьте установить библиотеку:

npm install react-bootstrap bootstrap

Разделим наше приложение на несколько компонентов, чтобы код был более организован и структурирован. Мы создадим следующие компоненты:

1. **NavbarComponent.js** - компонент для навигационной панели.
2. **MyCard.js** - компонент карточки.
3. **DataTable.js** - компонент таблицы данных.
4. **ModalComponent.js** - компонент модального окна.
5. **App.js** - главный компонент, который объединяет все остальные компоненты.

В папке **src/components**, создадим файл **NavbarComponent.jsx**, со следующим содержимым:

```
import React from 'react';
import { Navbar, Nav, Container } from 'react-bootstrap';

function NavbarComponent() {
  return (
    <Navbar bg="dark" variant="dark" expand="lg">
      <Container>
        <Navbar.Brand href="#home">My React App</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#link">Link</Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default NavbarComponent;
```

В папке **src/components**, создадим файл **MyCard.jsx**, со следующим содержимым:

```
import React from 'react';
import { Card, Button } from 'react-bootstrap';

function MyCard({ clickCount, onClick }) {
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Img variant="top" src="https://via.placeholder.com/150" />
      <Card.Body>
        <Card.Title>Card Title</Card.Title>
        <Card.Text>
          Some quick example text to build on the card title and make up the bulk
of the card's content.
        </Card.Text>
        <Button variant="primary" onClick={onClick}>
```

```
          Click Me ({clickCount})
        </Button>
      </Card.Body>
    </Card>
  );
}

export default MyCard;
```

Каждая карточка отображает кнопку с счетчиком кликов. При нажатии на кнопку значение счетчика увеличивается, и это значение передается в карточку через пропсы.

В папке **src/components**, создадим файл **DataTable.jsx**, со следующим содержимым:

```
import React from 'react';
import { Table } from 'react-bootstrap';

function DataTable({ data }) {
  return (
    <Table striped bordered hover>
      <thead>
        <tr>
          <th>#</th>
          <th>Text</th>
        </tr>
      </thead>
      <tbody>
        {data.map((item) => (
          <tr key={item.id}>
            <td>{item.id}</td>
            <td>{item.name}</td>
          </tr>
        ))}
      </tbody>
    </Table>
  );
}

export default DataTable;
```

Таблица отображает введенные пользователем данные. При каждом отправлении формы добавляется новая строка с текстом и номером записи.

В папке **src/components**, создадим файл **ModalComponent.jsx**, со следующим содержимым:

```jsx
import React, { useState } from 'react';
import { Modal, Button, Form } from 'react-bootstrap';

function ModalComponent({ onSubmit }) {
  const [show, setShow] = useState(false);
  const [inputValue, setInputValue] = useState('');

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  const handleSubmit = (e) => {
    e.preventDefault();
    onSubmit(inputValue);
    setInputValue('');
    handleClose();
  };

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Open Modal
      </Button>

      <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
          <Modal.Title>Enter Data</Modal.Title>
        </Modal.Header>
        <Modal.Body>
          <Form onSubmit={handleSubmit}>
            <Form.Group controlId="formBasicEmail">
              <Form.Label>Input Text</Form.Label>
              <Form.Control
                type="text"
                placeholder="Enter text"
                value={inputValue}
                onChange={(e) => setInputValue(e.target.value)}
              />
            </Form.Group>
            <Button variant="primary" type="submit">
              Submit
            </Button>
          </Form>
        </Modal.Body>
      </Modal>
    </>
  );
}

export default ModalComponent;
```

В модальном окне появилась форма, которая позволяет вводить текст. После нажатия на кнопку "Submit" текст отправляется в App, где обновляется таблица.

Перейдем в компонент **App.jsx** и определим следующее содержимое:

```
import React, { useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';

import NavbarComponent from './components/NavbarComponent';
import MyCard from './components/MyCard';
import DataTable from './components/DataTable';
import ModalComponent from './components/ModalComponent';

function App() {
  const [clickCount, setClickCount] = useState(0);
  const [modalData, setModalData] = useState('');
  const [tableData, setTableData] = useState([]);

  // Обработчик клика для увеличения счетчика
  const handleCardClick = () => {
    setClickCount(clickCount + 1);
  };

  // Обработчик для формы в модальном окне
  const handleModalSubmit = (data) => {
    setModalData(data);
    setTableData([...tableData, { name: data, id: tableData.length + 1 }]);
  };

  return (
    <>
      <NavbarComponent />
      <div className="container mt-4">
        <div className="row">
          <div className="col">
            <MyCard clickCount={clickCount} onClick={handleCardClick} />
          </div>
          <div className="col">
            <MyCard clickCount={clickCount} onClick={handleCardClick} />
          </div>
          <div className="col">
            <MyCard clickCount={clickCount} onClick={handleCardClick} />
          </div>
        </div>

        <div className="row mt-4">
          <div className="col">
            <h2>Data Table</h2>
            <DataTable data={tableData} />
          </div>
        </div>
```

```
        <div className="row mt-4">
          <div className="col text-center">
            <ModalComponent onSubmit={handleModalSubmit} />
          </div>
        </div>
      </div>
    </>
  );
}

export default App;
```

Таким образом, приложение становится интерактивным и позволяет пользователям взаимодействовать с компонентами, изменяя состояние и отображаемые данные.

React Bootstrap: Modals - полное взаимодействие

Модальные окна (Modals) - один из самых востребованных UI-компонентов. С их помощью удобно:

- просматривать детали элемента,
- подтверждать удаление,
- редактировать данные без перехода на другую страницу.

Если вы используете новый проект, не забудьте установить библиотеку:

npm install react-bootstrap bootstrap

Для начала, посмотрим простой пример по использованию модального коан с двумя кнопками. Изменим компонент **App.jsx**:

```
import "bootstrap/dist/css/bootstrap.min.css";
import ProductList from "./components/ProductList";

import { useState } from "react";
import { Button, Modal } from "react-bootstrap";

function App() {
  const [show, setShow] = useState(false);

  const handleOpen = () => setShow(true);
  const handleClose = () => setShow(false);

  const handleConfirm = () => {
    alert("Action confirmed");
    handleClose();
  };

  return (
```

```
    <div className="p-4">
      <h1>Modal example</h1>

      <Button variant="primary" onClick={handleOpen}>
        Open modal
      </Button>

      <Modal show={show} onHide={handleClose} centered>
        <Modal.Header closeButton>
          <Modal.Title>Confirmation</Modal.Title>
        </Modal.Header>

        <Modal.Body>Are you sure you want to continue?</Modal.Body>

        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Cancel
          </Button>
          <Button variant="danger" onClick={handleConfirm}>
            Confirm
          </Button>
        </Modal.Footer>
      </Modal>
    </div>
  );
}

export default App;
```

Компонент **App** использует состояние **show** для управления видимостью модального окна: при нажатии на кнопку оно открывается, а при закрытии или подтверждении - скрывается. Модальное окно из **react-bootstrap** рендерится контролируемо через проп **show**, автоматически обрабатывает клики по фону, кнопку закрытия и **Esc**, а действия пользователя обрабатываются через обычные обработчики событий React.

Давайте создадим приложение со списком продуктов, где все действия будут выполняться через модальные окна.

Каждое модальное окно определим в видео отдельного компонента. Структура проекта будет такой:

```
ProductList
  ├── ProductTable
  ├── ProductDetailsModal
  ├── ProductDeleteModal
  └── ProductEditModal
```

В папке **src/components**, определим компонент **ProductDetailsModal.jsx**:

```
import { Button, Modal } from "react-bootstrap";
```

```
export default function ProductDetailsModal({ show, product, onClose }) {
  return (
    <Modal show={show} onHide={onClose}>
      <Modal.Header closeButton>
        <Modal.Title>Product Details</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        {product && (
          <>
            <p>
              <strong>Name:</strong> {product.name}
            </p>
            <p>
              <strong>Price:</strong> ${product.price}
            </p>
            <p>
              <strong>Description:</strong> {product.description}
            </p>
          </>
        )}
      </Modal.Body>
      <Modal.Footer>
        <Button variant="secondary" onClick={onClose}>
          Close
        </Button>
      </Modal.Footer>
    </Modal>
  );
}
```

Компонент принимает три пропса: **show**, **product** и **onClose**:

- Проп **show** определяет, должно ли модальное окно быть открытым или закрытым.
- Проп **onClose** содержит функцию, которая вызывается при закрытии модального окна.
- Проп **product** содержит объект выбранного продукта, данные которого нужно отобразить.

Компонент **Modal** используется для создания модального окна Bootstrap.

- Атрибут **show={show}** управляет видимостью модального окна.
- Атрибут **onHide={onClose}** указывает, какая функция будет вызвана при попытке закрыть окно.
- Компонент **Modal.Header** отображает верхнюю часть модального окна.
- Атрибут **closeButton** добавляет кнопку закрытия в заголовок модального окна.

Компонент **Modal.Title** выводит заголовок модального окна с текстом «Product Details».
Компонент **Modal.Body** содержит основной контент модального окна. Отображение данных продукта происходит только в том случае, если объект **product** существует.

Условный рендеринг **product && (...)** предотвращает ошибки при отсутствии данных. Компонент **Modal.Footer** отображает нижнюю часть модального окна.

Компонент **Button** отображает кнопку с вариантом оформления secondary. При нажатии на кнопку вызывается функция **onClose**.

В папке **src/components**, определим компонент **ProductDeleteModal.jsx**:

```
import { Button, Modal } from "react-bootstrap";

export default function ProductDeleteModal({
  show,
  product,
  onClose,
  onConfirm,
}) {
  return (
    <Modal show={show} onHide={onClose}>
      <Modal.Header closeButton>
        <Modal.Title>Delete Product</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        Are you sure you want to delete
        <strong> {product?.name}</strong>?
      </Modal.Body>
      <Modal.Footer>
        <Button variant="secondary" onClick={onClose}>
          Cancel
        </Button>
        <Button variant="danger" onClick={onConfirm}>
          Delete
        </Button>
      </Modal.Footer>
    </Modal>
  );
}
```

Компонент принимает четыре пропса: **show**, **product**, **onClose** и **onConfirm**:

- Проп **show** определяет, отображается ли модальное окно.
- Проп **product** содержит объект продукта, который планируется удалить.
- Проп **onClose** содержит функцию для закрытия модального окна без выполнения действия.
- Проп **onConfirm** содержит функцию, которая выполняет удаление продукта.

Компонент **Modal** используется для создания модального окна подтверждения.

- Атрибут **show={show}** управляет видимостью модального окна.
- Атрибут **onHide={onClose}** указывает функцию, вызываемую при закрытии окна.

Компонент **Modal.Header** отображает заголовок модального окна. Атрибут **closeButton** добавляет кнопку закрытия в правой части заголовка.

Компонент **Modal.Title** выводит заголовок «Delete Product». Компонент **Modal.Body** содержит текст подтверждения удаления. В тексте отображается имя продукта с использованием опциональной цепочки **product?.name**. Опциональная цепочка предотвращает ошибку, если объект product отсутствует.

Компонент **Modal.Footer** содержит элементы управления действиями пользователя. Кнопка **Cancel** закрывает модальное окно без удаления продукта. Кнопка **Delete** вызывает функцию **onConfirm**.

В папке **src/components**, определим компонент **ProductEditModal.jsx**:

```
import { useEffect, useState } from "react";
import { Button, Form, Modal } from "react-bootstrap";

export default function ProductEditModal({ show, product, onClose, onSave }) {
  const [form, setForm] = useState({
    name: "",
    price: "",
    description: "",
  });

  useEffect(() => {
    if (product) {
      setForm(product);
    }
  }, [product]);

  const handleChange = (e) => {
    setForm({
      ...form,
      [e.target.name]: e.target.value,
    });
  };

  const handleSave = () => {
    onSave(form);
  };

  return (
    <Modal show={show} onHide={onClose}>
      <Modal.Header closeButton>
        <Modal.Title>Edit Product</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form>
          <Form.Group className="mb-3">
            <Form.Label>Name</Form.Label>
            <Form.Control
              name="name"
```

```
              value={form.name}
              onChange={handleChange}
            />
          </Form.Group>

          <Form.Group className="mb-3">
            <Form.Label>Price</Form.Label>
            <Form.Control
              type="number"
              name="price"
              value={form.price}
              onChange={handleChange}
            />
          </Form.Group>

          <Form.Group>
            <Form.Label>Description</Form.Label>
            <Form.Control
              as="textarea"
              rows={3}
              name="description"
              value={form.description}
              onChange={handleChange}
            />
          </Form.Group>
        </Form>
      </Modal.Body>
      <Modal.Footer>
        <Button variant="secondary" onClick={onClose}>
          Cancel
        </Button>
        <Button variant="success" onClick={handleSave}>
          Save changes
        </Button>
      </Modal.Footer>
    </Modal>
  );
}
```

Компонент принимает четыре пропса: **show**, **product**, **onClose** и **onSave**:

- Проп **show** управляет отображением модального окна редактирования.
- Проп **product** содержит объект продукта, который редактируется.
- Проп **onClose** содержит функцию для закрытия модального окна без сохранения изменений.
- Проп **onSave** содержит функцию для сохранения изменённых данных продукта.

Хук **useEffect** используется для синхронизации состояния формы с выбранным продуктом. Эффект срабатывает каждый раз, когда изменяется значение **product**. Если объект **product** существует, его данные копируются в состояние формы. Функция

handleChange обрабатывает изменения полей формы. При вводе данных обновляется соответствующее поле в состоянии **form**.

Функция **handleSave** вызывает функцию **onSave**, передавая обновлённый объект продукта. Компонент **Modal** используется для отображения формы в модальном окне:

- Атрибут **show={show}** управляет видимостью модального окна.
- Атрибут **onHide={onClose}** указывает функцию закрытия окна.

Кнопка **Cancel** закрывает модальное окно без сохранения изменений. Кнопка **Save** changes сохраняет изменения и имеет вариант оформления **success**.

В папке **src/components**, определим компонент **ProductTable.jsx**:

```
import { Button, Table } from "react-bootstrap";

export default function ProductTable({
  products,
  onDetails,
  onEdit,
  onDelete,
}) {
  return (
    <Table striped bordered hover>
      <thead>
        <tr>
          <th>Name</th>
          <th>Price ($)</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {products.map((product) => (
          <tr key={product.id}>
            <td>{product.name}</td>
            <td>{product.price}</td>
            <td className="d-flex gap-2">
              <Button variant="info" onClick={() => onDetails(product)}>
                Details
              </Button>
              <Button variant="warning" onClick={() => onEdit(product)}>
                Edit
              </Button>
              <Button variant="danger" onClick={() => onDelete(product)}>
                Delete
              </Button>
            </td>
          </tr>
        ))}
      </tbody>
    </Table>
  );
```

```
}
```

Компонент принимает четыре пропса: **products**, **onDetails**, **onEdit** и **onDelete**:

- Проп **products** содержит массив объектов продуктов для отображения в таблице.
- Проп **onDetails** содержит функцию, вызываемую при нажатии кнопки «Details».
- Проп **onEdit** содержит функцию, вызываемую при нажатии кнопки «Edit».
- Проп **onDelete** содержит функцию, вызываемую при нажатии кнопки «Delete».

Компонент **Table** создаёт таблицу с полосами, границами и эффектом наведения на строки. Атрибуты **striped**, **bordered** и **hover** задают стиль таблицы. Внутри таблицы создаётся **thead**, содержащий заголовки колонок: **Name, Price ($)** и **Actions**. Внутри **tbody** происходит итерация массива **products** с помощью метода **map**.

Кнопка «**Details**» имеет стиль info и вызывает функцию onDetails, передавая объект продукта. Кнопка «**Edit**» имеет стиль warning и вызывает функцию onEdit, передавая объект продукта. Кнопка «**Delete**» имеет стиль danger и вызывает функцию onDelete, передавая объект продукта.

В папке **src/components**, определим компонент **ProductList.jsx**:

```
import { useState } from "react";
import ProductTable from "./ProductTable";
import ProductDetailsModal from "./ProductDetailsModal";
import ProductDeleteModal from "./ProductDeleteModal";
import ProductEditModal from "./ProductEditModal";

const initialProducts = [
  {
    id: 1,
    name: "Laptop",
    price: 1200,
    description: "Powerful laptop for work",
  },
  {
    id: 2,
    name: "Smartphone",
    price: 800,
    description: "Flagship mobile device",
  },
  { id: 3, name: "Headphones", price: 150, description: "Wireless headphones" },
];

export default function ProductList() {
  const [products, setProducts] = useState(initialProducts);
  const [selectedProduct, setSelectedProduct] = useState(null);

  const [showDetails, setShowDetails] = useState(false);
  const [showDelete, setShowDelete] = useState(false);
  const [showEdit, setShowEdit] = useState(false);
```

```
const openDetails = (product) => {
  setSelectedProduct(product);
  setShowDetails(true);
};

const openDelete = (product) => {
  setSelectedProduct(product);
  setShowDelete(true);
};

const openEdit = (product) => {
  setSelectedProduct(product);
  setShowEdit(true);
};

const deleteProduct = () => {
  setProducts(products.filter((p) => p.id !== selectedProduct.id));
  setShowDelete(false);
};

const saveEdit = (updatedProduct) => {
  setProducts(
    products.map((p) => (p.id === updatedProduct.id ? updatedProduct : p))
  );
  setShowEdit(false);
};

return (
  <>
    <h2 className="mb-4">Product List</h2>

    <ProductTable
      products={products}
      onDetails={openDetails}
      onEdit={openEdit}
      onDelete={openDelete}
    />

    <ProductDetailsModal
      show={showDetails}
      product={selectedProduct}
      onClose={() => setShowDetails(false)}
    />

    <ProductDeleteModal
      show={showDelete}
      product={selectedProduct}
      onClose={() => setShowDelete(false)}
      onConfirm={deleteProduct}
    />
```

```
      <ProductEditModal
        show={showEdit}
        product={selectedProduct}
        onClose={() => setShowEdit(false)}
        onSave={saveEdit}
      />
    </>
  );
}
```

Внутри компонента создаётся состояние **products** с начальным значением **initialProducts**. Состояние **selectedProduct** хранит объект продукта, который выбран для просмотра, редактирования или удаления. Состояния **showDetails**, **showDelete** и **showEdit** управляют видимостью соответствующих модальных окон.

- Функция **openDetails** устанавливает выбранный продукт и открывает модальное окно с деталями.
- Функция **openDelete** устанавливает выбранный продукт и открывает модальное окно удаления.
- Функция **openEdit** устанавливает выбранный продукт и открывает модальное окно редактирования.
- Функция **deleteProduct** удаляет выбранный продукт из массива products и закрывает модальное окно удаления.
- Функция **saveEdit** обновляет данные продукта в массиве products и закрывает модальное окно редактирования.

Все модальные окна получают объект selectedProduct и функции закрытия или сохранения изменений через соответствующие пропсы. Этот компонент реализует полный CRUD-интерфейс для списка продуктов с использованием таблицы и модальных окон.

Изменим компонент **App.jsx**:

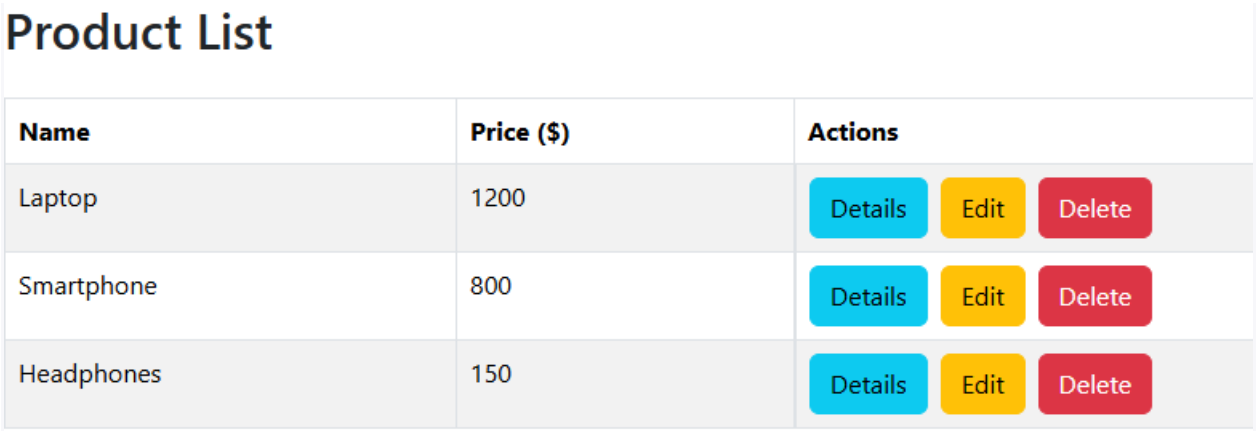
```
import "bootstrap/dist/css/bootstrap.min.css";
import ProductList from "./components/ProductList";

function App() {
  return (
    <div>
      <ProductList />
    </div>
  );
}

export default App;
```

В компонент обязательно подключаем стили **Bootstrap**.

Запустим приложение и проверим его работу:



Списка

Списка

списка

Приложение – «Список покупок»

Давайте создадим приложение для списка покупок с использованием React и компонентов Bootstrap из библиотеки react-bootstrap. Если вы используете новый проект, не забудьте установить библиотеку:

npm install react-bootstrap bootstrap

Разделим наше приложение на несколько компонентов, чтобы код был более организован и структурирован. Мы создадим следующие компоненты:

- **ShoppingList** – отображает все покупки с пагинацией.
- **SearchBar** – предоставляет возможность поиска по списку покупок.
- **AddEditModal** – модальное окно для добавления и редактирования покупок.
- **ConfirmModal** – модальное окно для подтверждения удаления покупки.
- **App** – главный компонент, который объединяет все компоненты вместе.

В папке **src/components**, создадим файл **ShoppingList.jsx**, со следующим содержимым:

```
import React, { useState } from 'react';
import { ListGroup, Button, Pagination } from 'react-bootstrap';

const ITEMS_PER_PAGE = 5;

function ShoppingList({ items, onEdit, onDelete }) {
  const [currentPage, setCurrentPage] = useState(1);

  // Вычисление начального и конечного индекса элементов на текущей странице
  const startIndex = (currentPage - 1) * ITEMS_PER_PAGE;
  const endIndex = startIndex + ITEMS_PER_PAGE;
```

```
  const paginatedItems = items.slice(startIndex, endIndex);

  const totalPages = Math.ceil(items.length / ITEMS_PER_PAGE);

  return (
    <>
      <ListGroup className="mt-4">
        {paginatedItems.map((item) => (
          <ListGroup.Item key={item.id}>
            {item.name}
            <Button
              variant="info"
              size="sm"
              className="float-end ms-2"
              onClick={() => onEdit(item)}
            >
              Edit
            </Button>
            <Button
              variant="danger"
              size="sm"
              className="float-end"
              onClick={() => onDelete(item)}
            >
              Delete
            </Button>
          </ListGroup.Item>
        ))}
      </ListGroup>
      <Pagination className="mt-4">
        {Array.from({ length: totalPages }, (_, index) => (
          <Pagination.Item
            key={index + 1}
            active={index + 1 === currentPage}
            onClick={() => setCurrentPage(index + 1)}
          >
            {index + 1}
          </Pagination.Item>
        ))}
      </Pagination>
    </>
  );
}

export default ShoppingList;
```

В папке **src/components**, создадим файл **SearchBar.jsx**, со следующим содержимым:

```
import React from 'react';
```

```
import { Form } from 'react-bootstrap';

function SearchBar({ setSearchTerm }) {
  return (
    <Form.Control
      type="text"
      placeholder="Search items..."
      className="mt-4"
      onChange={(e) => setSearchTerm(e.target.value)}
    />
  );
}

export default SearchBar;
```

В папке **src/components**, создадим файл **AddEditModal.jsx**, со следующим содержимым:

```
import React, { useState, useEffect } from 'react';
import { Modal, Button, Form } from 'react-bootstrap';

function AddEditModal({ show, onHide, item, onSave }) {
  const [name, setName] = useState('');

  useEffect(() => {
    if (show) {
      if (item) {
        setName(item.name);
      } else {
        setName('');
      }
    }
  }, [show, item]);

  const handleSubmit = (e) => {
    e.preventDefault();
    if (name.trim()) {
      onSave({ id: item ? item.id : null, name: name.trim() });
      handleClose();
    }
  };

  const handleClose = () => {
    setName(''); // Reset the name field
    onHide();
  };

  return (
    <Modal show={show} onHide={handleClose}>
```

```
      <Modal.Header closeButton>
        <Modal.Title>{item ? 'Edit Item' : 'Add New Item'}</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form onSubmit={handleSubmit}>
          <Form.Group controlId="formItemName">
            <Form.Label>Item Name</Form.Label>
            <Form.Control
              type="text"
              placeholder="Enter item name"
              value={name}
              onChange={(e) => setName(e.target.value)}
              required
            />
          </Form.Group>
          <Button variant="primary" type="submit" className="mt-3">
            Save
          </Button>
        </Form>
      </Modal.Body>
    </Modal>
  );
}

export default AddEditModal;
```

В папке **src/components**, создадим файл **ConfirmModal.jsx**, со следующим содержимым:

```
import React from 'react';
import { Modal, Button } from 'react-bootstrap';

function ConfirmModal({ show, onHide, onConfirm }) {
  return (
    <Modal show={show} onHide={onHide}>
      <Modal.Header closeButton>
        <Modal.Title>Confirm Deletion</Modal.Title>
      </Modal.Header>
      <Modal.Body>Are you sure you want to delete this item?</Modal.Body>
      <Modal.Footer>
        <Button variant="secondary" onClick={onHide}>
          Cancel
        </Button>
        <Button variant="danger" onClick={onConfirm}>
          Delete
        </Button>
      </Modal.Footer>
    </Modal>
  );
```

```
}

export default ConfirmModal;
```

Перейдем в компонент **App.jsx** и определим следующее содержимое:

```
import React, { useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import { Container, Button } from 'react-bootstrap';
import ShoppingList from './components/ShoppingList';
import SearchBar from './components/SearchBar';
import AddEditModal from './components/AddEditModal';
import ConfirmModal from './components/ConfirmModal';

function App() {
  const [items, setItems] = useState([
    { id: 1, name: 'Milk' },
    { id: 2, name: 'Bread' },
    { id: 3, name: 'Eggs' },
    { id: 4, name: 'Butter' },
  ]);
  const [searchTerm, setSearchTerm] = useState('');
  const [modalShow, setModalShow] = useState(false);
  const [editingItem, setEditingItem] = useState(null);
  const [confirmModalShow, setConfirmModalShow] = useState(false);
  const [itemToDelete, setItemToDelete] = useState(null);

  // Функция для добавления или обновления элемента
  const handleSaveItem = (item) => {
    if (editingItem) {
      setItems(items.map((i) => (i.id === editingItem.id ? item : i)));
    } else {
      setItems([...items, { ...item, id: items.length + 1 }]);
    }
    setModalShow(false);
  };

  // Функция для удаления элемента
  const handleDeleteItem = () => {
    setItems(items.filter((i) => i.id !== itemToDelete.id));
    setConfirmModalShow(false);
  };

  // Фильтр элементов по строке поиска
  const filteredItems = items.filter((item) =>
    item.name.toLowerCase().includes(searchTerm.toLowerCase())
  );

  return (
```

```
    <Container>
      <h1 className="mt-4">Shopping List</h1>
      <SearchBar setSearchTerm={setSearchTerm} />
      <Button variant="primary" onClick={() => {
        setEditingItem(null);
        setModalShow(true);
      }}>
        Add New Item
      </Button>
      <ShoppingList
        items={filteredItems}
        onEdit={(item) => {
          setEditingItem(item);
          setModalShow(true);
        }}
        onDelete={(item) => {
          setItemToDelete(item);
          setConfirmModalShow(true);
        }}
      />
      <AddEditModal
        show={modalShow}
        onHide={() => setModalShow(false)}
        item={editingItem}
        onSave={handleSaveItem}
      />
      <ConfirmModal
        show={confirmModalShow}
        onHide={() => setConfirmModalShow(false)}
        onConfirm={handleDeleteItem}
      />
    </Container>
  );
}

export default App;
```

Теперь у нас есть полностью функциональное приложение для списка покупок, использующее компоненты Bootstrap и обеспечивающее основные действия, такие как отображение, поиск, добавление, редактирование и удаление элементов.